C Programming

Class- BCA IInd Semester

Dr. Dharm Raj Singh

Assistant Professor, (HOD)
Department of Computer Application
JagatpurP. G. College, Varanasi
Affiliated to Mahatma Gandhi Kashi Vidhyapith Varanasi
Email- dharmrajsingh67@yahoo.com

Outline

unit 1 : Array

- Initialization of One Dimensional Array
- Multi Dimensional Array:
- Multi Dimensional Array:
- Initializing Two-Dimensional Arrays:
- Accessing Two-Dimensional Array Elements:

Grouping Data (Array)

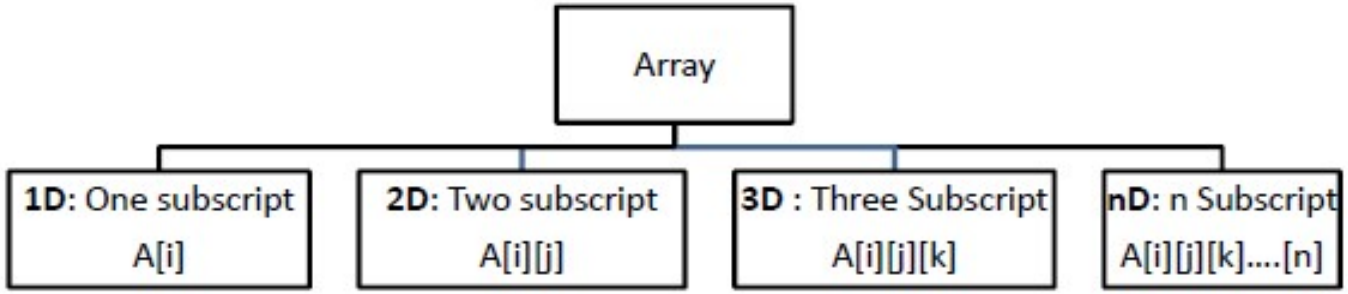
ARRAY: - Concise and an efficient representation of related data items.
- An Array is a group of related data items that share a common name.

Price of *n* items: Shares a common attribute that they are numeric value.

	item1p	item2p	item3p	item4p	item5p	item6p	item7p	item8p	-------	itemnp
Index	0	1	2	3	4	5	6	7	-------	n

This can be represented using a common name (say price) and index / subscript, as different terms represented in a sequence.

e.g. price[5] gives the price of the fifth item in the list.



Grouping Data (1D Array)

1D ARRAY: List of items given one variable name using only one subscript and such a variable is called a *single-subscripted variable* or a one –dimensional variable.

- In mathematics, single subscripted variable such as x_i .
- In C, single subscripted variable x_i can be expressed as x[0], x[1], x[2], ..., x[n].

Declaration of 1D Array:

type *variable-name*[size];

- *type* specifies the type of element that will be contained in the array, such as **int**, **float,** or **char** .
- ***size*** indicates the maximum number of elements that can be stored inside the array

```
float height[50];  % declares the height to be an array containing 50 real elements.
int group[10];     % declares the group as an array to contain a maximum of 10 integer
                   % constant.
```

- C language treats the ***character string*** as arrays of characters. The ***size*** in a character string represents the maximum number of characters that the string can hold.

Initialization of One Dimensional Array

- Once an array is declared, it must be initialized. Otherwise array will contain the garbage values. There are two different ways in which we can initialize the static array

1. Compile time
2. Run time

- **Compile Time Initialization**

data-type array-name[size] = { list of values separated by comma };

For example:

int var[5] = {5, 7, 9, 3, 4};

- **Run Time Initialization**

An array can initialized at run time by the program or by taking the input from the keyboard. Generally, the large arrays are declared at run time in the program itself. Such as –

```
main()
{
  int i;
  int x[10];
  for (i=0; i<10; i++)
 {
    x[i] = i + 1;
 printf( “x=%d.\n", x[i]);
  }
 }
```

```
main( )
{
int val[10], i, total=0;
printf(“Enter any ten numbers: ”);
for(i=0;i<10;i++)
scanf(“%d”, &val[i]);
for(i=0;i<10;i++)
total = total + val[i];
printf(“\nTotal is: %d”, total);
}
```

Classification of Array: Array are divided into two part-

- **One Dimensional Array**
- **Multi Dimensional Array**

One Dimensional Array: Dimension is nothing but no of pairs of square brackets placed after the name of the array. It can be defined as one dimensional array has only one subscript specification is required for specify a particular element of an array. The syntax of the declaring one dimensional array is as follows:

Data_Type Array_Name[Expression];

For Example: int marks[10];

Where int is the type of the array, marks is name of the array and expression [10] is shows the maximum number of the element in the array.

Write a C program to print Element of Array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i, array[30], n;
  printf("\n Enter the no of elements :");
  scanf("%d", &n);
  printf("\n Enter the values :");            //Read values into Array
  for (i = 0; i < n; i++)
 {
              scanf("%d", &array[i]);
 }
  for (i = 0; i < n; i++)                                    //Printing of all elements of array
 {
    printf("\n array[%d] = %d", i, array[i]);
  }
 getch( );
}
```

Multi Dimensional Array: It can be defined as 'Array which holds more than one subscript is known as Multi Dimensional Array. Generally 2-D array is called as Matrix. It is more suitable for the processing of table and matrix manipulations. C language allow to programmers to use arrays with more than two dimensions. On behalf of this it can be divided into two sub section:

Two Dimensional Array

N-Dimensional Array

Data_Type Array_Name[row size][column size];

For Example: int n[3][4];

Where int is the type of the array, n is the array name and row size is the number of rows and column size the number of column. To find the total number of element of an array, multiply the total number of row with the total number of column.

For Example: int a[3][4];

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Thus, every element in array a is identified by an element name of the form a[i][j], where a of the array, and i and j are the subscripts (index) that uniquely identify each element in a.

Initializing Two-Dimensional Arrays:

- Multidimensional arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row has 4 columns.

```
int a[3][4] = {
 {0, 1, 2, 3} ,   /*  initializers for row indexed by 0 */
 {4, 5, 6, 7} ,   /*  initializers for row indexed by 1 */
 {8, 9, 10, 11}   /*  initializers for row indexed by 2 */
};
```

Accessing Two-Dimensional Array Elements:

An element in 2-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example:

```
int val = a[2][3];
```

The above statement will take 4th element from the 3rd row of the array.

Example

```
#include <stdio.h>
int main ()
{
int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}}; /* an array with 5 rows and 2 columns*/
int i, j;
/* output each array element's value */
for ( i = 0; i < 5; i++ )
{
for ( j = 0; j < 2; j++ )
{
printf("a[%d][%d] = %d\n", i,j, a[i][j] );
}
}
return 0;
}
```

Write a program to print a matrix of order m * n where

```
void main( )
{
        clrscr( );
        int mat[50][50] i, j, m, n;
        printf("Enter the number of Rows");
        scanf("%d",&m);
        printf("Enter the number of Columns");
        scanf("%d",&n);
        printf("Enter the element for the Matrix");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        scanf("%d", &mat[i][j]);
                }
        }
        printf("The Matrix is:");
        for(i=0;i<m;i++)
        {
                for(j=0;j<n;j++)
                {
                        printf("\t%d",mat[i][j]);
                }
                printf("\n");
        }
        getch( );
}
```

Exercise

1. Write a program in C to merge two arrays of same size sorted in descending order.
2. Write a program in C to print all unique elements in an array.
3. Write a program in C to read n number of values in an array and display it in reverse order.
4. Write a program in C to find the sum of all elements of the array.
5. Write a program in C to count a total number of duplicate elements in an array.
6. Write a program in C to find the maximum and minimum element in an array.
7. Write a program in C to separate odd and even integers in separate arrays
8. Write a program in C for a 2D array of size 3x3 and print the matrix.
9. Write a program in C for addition of two Matrices of same size.
10. Write a program in C for multiplication of two square Matrices.
11. Write a program in C to find transpose of a given matrix.
12. Write a program in C to calculate determinant of a 3 x 3 matrix.
13. Write a program in C to generate a random permutation of array elements.
14. Write a program that interchanges the odd and even components of an array.

References

- Kanetkar, Yashavant P. "Let UsC Fifth Edition." (2017).
- Kernighan, Brian W., and Dennis M. Ritchie. *The C programming language*. 2006.
- Ritchie, Dennis M., Brian W. Kernighan, and Michael E. Lesk. The C programming language. Englewood Cliffs: Prentice Hall, 1988.
- McGraw-Hill, Herbert Schildt Tata. "The Complete Reference C fourth Edition”." (2005).
- Griffiths, David, and Dawn Griffiths. Head First C: A Brain-Friendly Guide. " O'Reilly Media, Inc.", 2012.
- Programming in C-Balguruswamy
- Structured programming approach using C-Forouzah & Ceilberg Thomson learning Publication

Declaration

“The content is exclusively meant for academic purpose and for enhancing teaching and learning. Any other use for economic/commercial purpose is strictly prohibited. The users of the content shall not distribute, disseminate or share it with anyone else and its use is restricted to advancement of individual knowledge. The information provided in this e-content is authentic and best as per knowledge”.

Dr. Dharm Raj Singh
Assistant Professor, (HOD)
Department of Computer Application
JagatpurP. G. College, Varanasi

Thanks